
IE 7275: Data Mining in Engineering

# Boston Airbnb Price Prediction

**Group 16**

Ajay Kartik Krishnakumar (NU ID : 001094072)

krishnakumar.aj@northeastern.edu

Vishwajit Chaure (NU ID : 001548283)

chaure.v@northeastern.edu

Instructor: Prof. Srinivasan Radhakrishnan

Submission Date: 08/20/2021

# INDEX

## Problem Setting:

Since its inception, Airbnb has been a critical market driver in addressing a low-cost lodging problem by investigating the ancillary revenue stream of residents who could not afford their apartment's rent. Airbnb now provides a solid, intermediate public framework for connecting supply and demand in the temporary housing and accommodation market. We wish to explore the underlying facts behind the listings of the various properties inside Boston and try to answer specific issues that could assist both Airbnb and potential owners as Airbnb has grown in popularity.

## Problem Definition:

We plan to investigate it from a distinct perspective to see if there is anything we can do to assist property owners in determining whether to advertise their home and what we can do to help them obtain the highest price, which would benefit both the landlord and Airbnb. This project aims to solve this problem, by using machine learning and data mining techniques to predict the base price for properties in Boston.

## Data Sources:

The Dataset used for this project is available on “insideairbnb.com”.

## Data Description:

The dataset consists of 74 columns and approximately 3146 rows.

Dataset Name: listings.csv

Few important variables:

- Accommodates: the number of guests the rental can accommodate.
- Bedrooms: number of bedrooms included in the rental.
- Bathrooms: number of bathrooms included in the rental.
- Price: nightly price for the rental.
- First Review: the date of the first review.
- Last Review: the date of the most recent review.
- Review Score Rating: guests can score properties overall from 1 to 5 stars

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3146 entries, 0 to 3145
Data columns (total 74 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   id                                            3146 non-null   int64
 1   listing_url                                   3146 non-null   object
 2   scrape_id                                     3146 non-null   int64
 3   last_scraped                                  3146 non-null   object
 4   name                                          3146 non-null   object
 5   description                                   3111 non-null   object
 6   neighborhood_overview                         2132 non-null   object
 7   picture_url                                   3146 non-null   object
 8   host_id                                       3146 non-null   int64
 9   host_url                                      3146 non-null   object
 10  host_name                                     3068 non-null   object
 11  host_since                                    3068 non-null   object
 12  host_location                                 3065 non-null   object
 13  host_about                                    2001 non-null   object
 14  host_response_time                            2452 non-null   object
 15  host_response_rate                            2452 non-null   object
 16  host_acceptance_rate                          2466 non-null   object
 17  host_is_superhost                             3068 non-null   float64
 18  host_thumbnail_url                            3068 non-null   object
 19  host_picture_url                              3068 non-null   object
 20  host_neighbourhood                            2873 non-null   object
 21  host_listings_count                           3068 non-null   float64
 22  host_total_listings_count                     3068 non-null   float64
 23  host_verifications                            3146 non-null   object
 24  host_has_profile_pic                          3068 non-null   float64
```

Fig 1: Variable Description

## Data Exploration:

- Checking whether boolean and categorical features contain enough instances in each category to make them worth including. It can be seen that several columns only contain one category and can be dropped while Pre-processing.
- The columns like ‘calculated host listings count shared rooms’, ‘review scores location’, ‘calendar updated’ has few values in it. These columns will be dropped during pre-processing. (Fig 2)

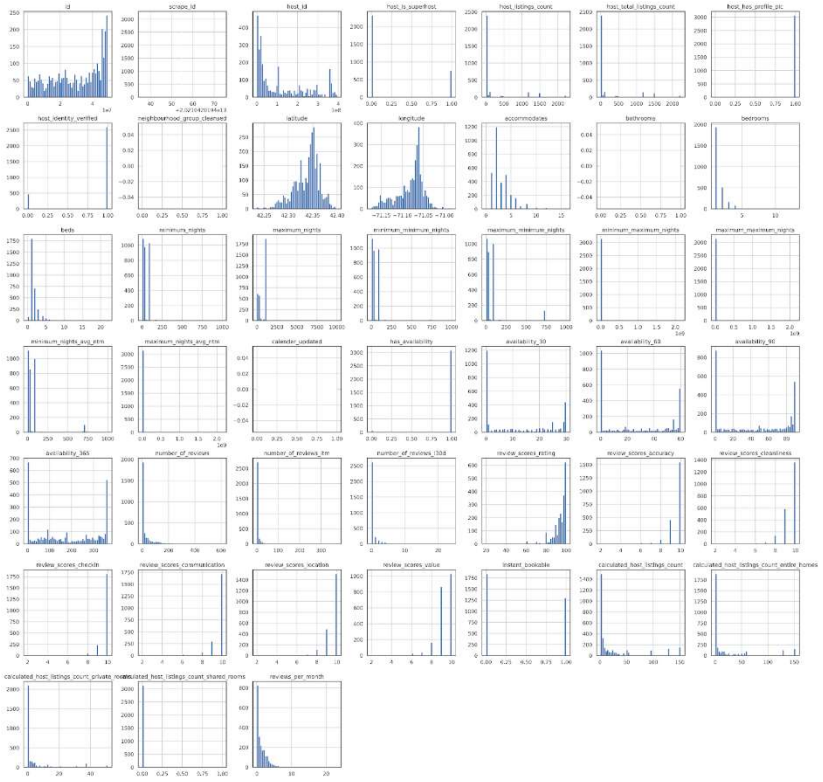


Fig 2: Boolean and Categorical variables plot

- These are text columns with description which will not be useful for prediction. Also, few columns like 'neighbourhood_group_cleansed', 'bathrooms', 'calendar_updated' have more than 75% missing values. (Fig 3)

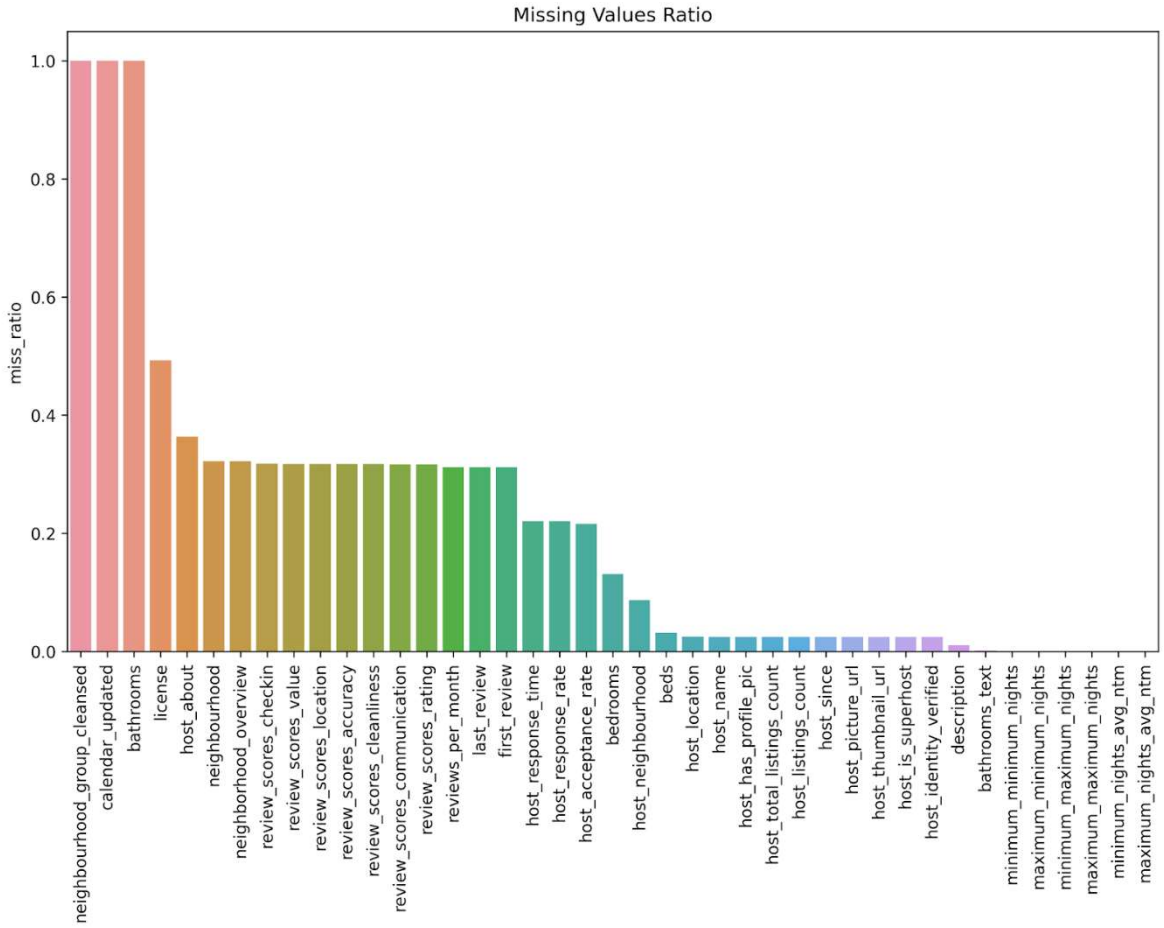



Fig 3: Missing values ratio

- The features ‘accommodates’, ’bedrooms’ and ‘beds’ are highly correlated with the target variable ‘price’. So, during pre-processing we can keep only one of these columns. (Fig 3)

- From Fig 4, It can observe that ‘Dorchester’ neighborhood has the most listings. We can also say that Boston has most listing in central area because neighborhoods like Downtown, South End, Back Bay, South Boston are among top ten neighborhoods having most listings.

- In Fig 5, We think the peak around late 2020 for 'Host joining Airbnb' might be after relaxation of covid restrictions. With the same reasoning, we can see that there is significant increase in listing getting first review.


Fig 4: Top 10 High Correlated Features with Target Feature 'Price'

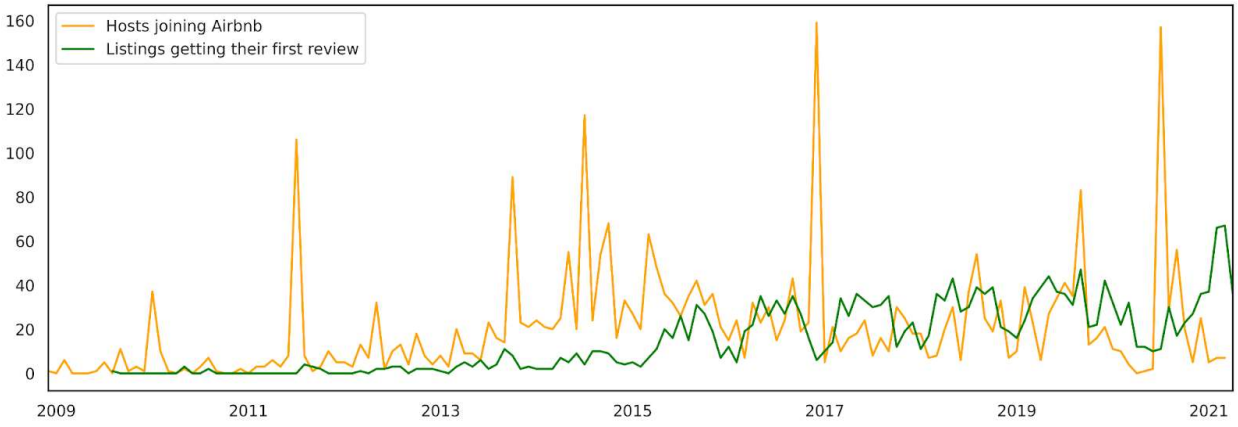



Fig 5: Time Series comparing Host Joining & First Review

## Data Mining Tasks:

### Data Pre-processing

- The first task we performed here was finding out the number of missing values in the dataset and then imputing them. We then imputed the percentage of missing values in each column and decided to get rid of columns which has more than 70% of missing values.

- NLP will not be used in the creation of an initial model (although they could be used to augment the model later, e.g., through sentiment analysis). Therefore, free text columns will be dropped for now, as will other columns which are not useful for predicting price (e.g., URL, Host name and other host-related features that are unrelated to the property).

- The 'host_listings_count' and 'host_total_listings_count' are the same in all but 78 cases. These cases are those where the value is NaN. Therefore, one of these columns can be dropped. Other columns which split these into type of property will also be dropped, as they will be highly correlated (one will be the total of the others).

- There are multiple columns for property location, including an attempt by the site that originally scraped the data to clean up the neighborhood locations. Some of these columns can be dropped. Because all the listings are in Boston, columns relating to city and country can be dropped. One column for area will be kept 'neighborhood cleansed'.
- There are multiple columns for minimum and maximum night stays, but the two main ones will be used as there are few differences between e.g., minimum_nights and minimum_minimum_nights. The latter presumably refers to the fact that min/max night stays can vary over the year. The default (i.e., most frequently applied) min/max night stay values will be used instead.

- The 'Host Since' is a datetime column and will be converted into a measure of the number of days that a host has been on the platform, measured from the date that the data was scraped (30 April 2021). The 'Host Since' column is dropped here.

- In ‘First Review’ and ‘Last Review’, About a quarter of listings have not had a review written for them. This is too large a proportion of the dataset to drop, and dropping the columns would lose a lot of useful information - reviews are very important in people's decisions to book, and therefore price.

  This is also too large a proportion of the dataset to simply replace with median/mean values, as this would skew the distribution substantially. Also, the missing values here are not really missing values, as the fact that they are NaNs is meaningful - it tells us that these are new or previously unbooked listings that have not had reviews yet. In order to make the resulting model work able to predict prices for any Airbnb listing, including brand new listings, is actually beneficial to keep them in. Therefore, these will be kept as an 'unknown' category, and the feature will have to be treated as categorical (and therefore one-hot encoded) rather than numerical.

Data Transformation

- In Property type, some cleaning of property types is required as there are many categories with only a few listings. The categories 'apartment', 'house' and 'other' will be used, as most properties can be classified as either apartments or houses.

- In Amenities, some amenities are more important than others (e.g., a balcony is more likely to increase price than a fax machine), and some are likely to be uncommon (e.g., 'Electric profiling bed'). Based on previous experience working in the Airbnb property management industry, and research into which amenities are considered by guests to be more important, a selection of the more important amenities will be extracted. For example, if it turns out that almost all properties have/do not have a particular amenity, that feature will not be very useful in helping explain differences in prices.

- For Host_response_rate, with about a third of values being null. This will also be kept as its own category, after grouping other values into meaningful groups (i.e., transforming this into a categorical feature, rather than a numerical one). Because about 70% of hosts respond 100% of the time, this will be kept as its own category, and other values will be grouped into bins.

Encoding and Standardization

- One-Hot encoding is used on all the categorical variables so that they can be used for modeling later.
- As numerical differences have scale differences applying standardization is necessary for model performance. We chose standardization over normalization because standardization is not affected by outliers.

## Data Mining Models/Methods:

Train Test Split

- We used Hold out method for train test split with 70% as training data and 30% as testing data.

**KNN**

- With Hyper-Parameter Tuning, parameters like Neighbors, Metric and Weights were tuned. We varied Neighbors in range of 1 to 21, metric between Euclidean and Manhattan and weight between uniform and distance.
- We also used KNN regressor with cross validation, Grid Search with Cross validation, Randomized search with cross validation to find the best parameters for regressor.
- Using applied PCA on scaled data, but it was not optimal result in terms of RMSE.

**Decision Tree**

- In Decision tree, we tuned parameters like max_depth and criterion to find the best performing model. The max_depth ranging from 1 to 31 and criterion for quality of split was varied among mse, friedman_mse and mae.

**Random Forest**

- For Random Forest, we tuned parameters like max_depth, criterion, and n_estimators. The parameter max_depth was tuned in the range of 1 to 31, criterion for data split quality mse and mae.

**SVR**

- For SVR, we tuned parameters like C regularization parameter, loss, dual and tol using Grid search cross validation. The parameters were varied as following.

  'C': [0.1, 1, 10, 100, 1000],
  'loss': ['epsilon_insensitive', 'squared_epsilon_insensitive','standard SVR'],
  'dual': [True, False],
  'tol': [0.0001, 0.00001]

**Neural Networks**

- For NN, we tuned parameters like learning_rate_init, transfer_function. The parameters were varied as following.

  learning_rate_init = [0.00001, 0.0001, 0.01, 0.03, 1, 3, 10]
  transfer_function = ['identity', 'logistic', 'tanh', 'relu']

**Ridge Regression**

- For Ridge, we tuned the alpha parameter. The parameters were varied as following.

alpha = [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 250, 500, 750, 1000, 1500, 2500, 5000 , 10000, 100000, 1000000]

**XG Boost**

- For XGB, we tuned parameters like learning rate, max depth, min child weight, subsample, colsample bytree, nestimators and objective. The parameters were varied as following.

  'learning rate': [0.01, 0.1],
  'max_depth': [3, 5, 7, 10],
  'min_child_weight': [1, 3, 5],
  'subsample': [0.5, 0.7],
  'colsample_bytree': [0.5, 0.7],
  'n_estimators' : [100, 200, 500],
  'objective': ['reg:squarederror']

## Performance Evaluation:

### KNN

Best performing model

| | #neighbors | distance | weight | rmse |
|---|---|---|---|---|
| 39 | 10 | manhattan | distance | 112.625 |

Fig 7: KNN – Best Performing Model

MAE: 52.70539330461238
RMSE: 112.62482256842674

Fig 8: KNN – MAE and RMSE

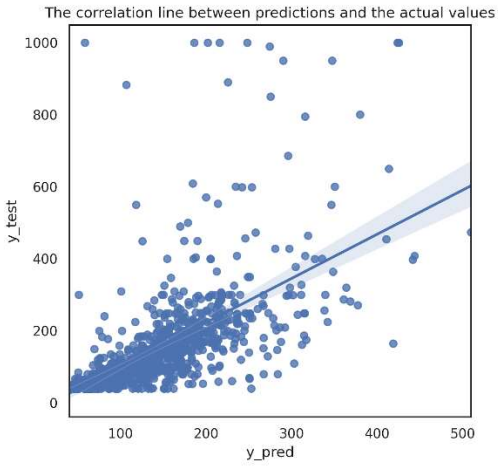



Fig 9: KNN – Correlation line between Predicted and Actual Values

- With Hyperparameter tuned KNN model, the best performing model was with neighbors as 10, distance as Manhattan and weight as distance. (Fig 7)
- The RMSE for best performing model is 112.625 and MAE is 52.70. (Fig 8)
- Fig 9 shows the correlation line between actual test values and predicted values.

### Decision Tree

- With Hyperparameter tuned Decision Tree, the best performing model was with depth as 5, purity method as MAE having RMSE 109.158 (Fig 10)
- From Fig 11, we can see that purity methods MAE has the least RMSE value as compared to MSE and Friedman_ MSE.

Best performing model

| | depth | puritymethod | rmse |
|---|---|---|---|
| 15 | 5 | mae | 109.158 |

Fig 10: Decision Tree – Best Performing Model

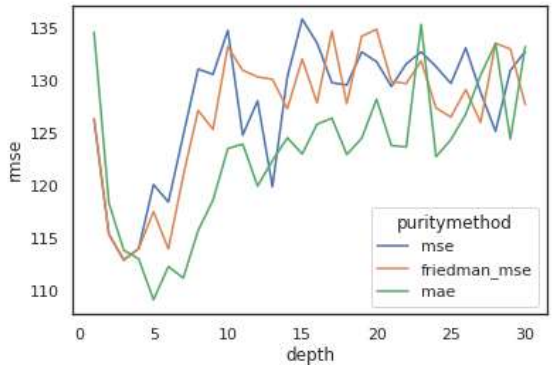



Fig 11: Decision Tree – Purity Method Plot

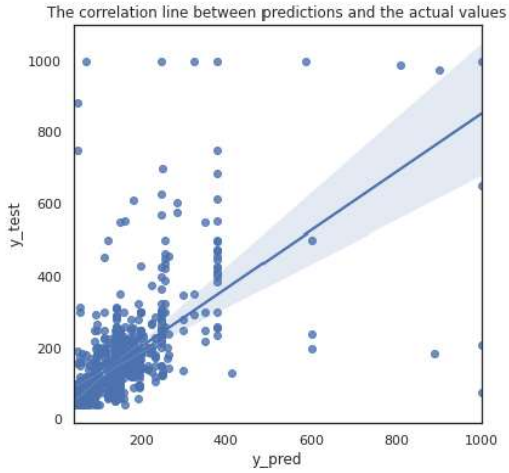



Fig 12: Decision Tree – Correlation Plot

- The Fig 12 shows correlation line plot between the Actual values and the predicted values.

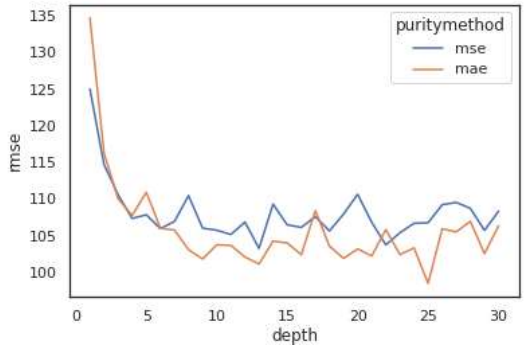
**Random Forest**

- For Random Forest, we opted to go with bagging. The tuned hyper parameters are depth and purity method.

Best performing model

| | depth | puritymethod | rmse |
|---|---|---|---|
| 50 | 25 | mae | 98.5266 |

Fig 13: Random Forest – Best Performing Model



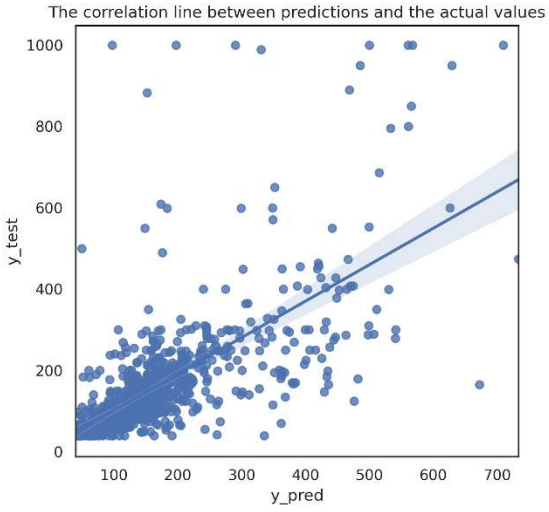




Fig 14: Random Forest – Purity Method Plot  Fig 15: Random Forest – Correlation Plot

- The best model was with depth as 25 and purity method as MAE having RMSE as 98.5266 (Fig 13).

**SVR**

- For SVR, we tuned parameters like C regularization parameter, loss, dual and tol using Grid search cross validation. The parameters were varied as following.

  ‘C': [0.1, 1, 10, 100, 1000] 'dual': [True, False]  ‘tol': [0.0001, 0.00001]

  'loss': ['epsilon_insensitive', 'squared_epsilon_insensitive','standard SVR']

```
MAE:  52.007492944251844
RMSE:  113.12868870992631
```

Fig 16: SVR – MAE and RMSE

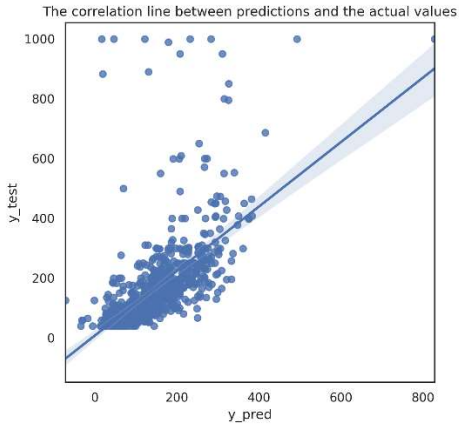



Fig 17: SVR – Correlation Plot

- The best SVR model has parameters C regularization as 100 and loss as epsilon insensitive.

**Neural Networks**

- After hyper parameter tuning, Neural Network regressor with learning rate 0.01 and transfer function as logistic gave the best RMSE score 110.218 (Fig 18).

Best performing model

| | Learning Rate | Transfer Function | RMSE |
|---|---|---|---|
| 9 | 0.01 | logistic | 110.218 |

Fig 18: NN – Best Performing Model

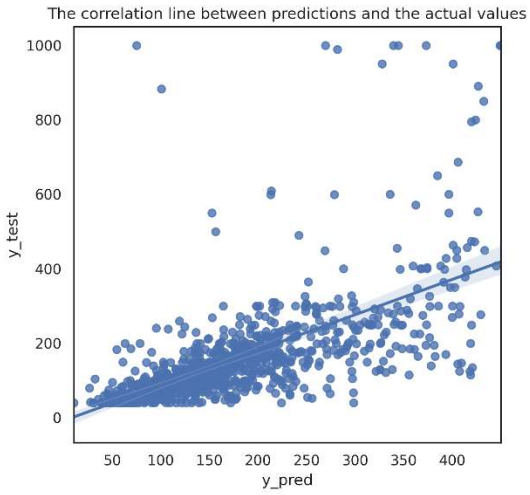



Fig 19: NN – Correlation Plot

**Ridge**

```
MAE:  52.70539330461238
RMSE:  106.71339299015915
```

Fig 20: Ridge – MAE and RMSE

- The best RMSE result for ridge regression was with hyper parameter alpha as 500.

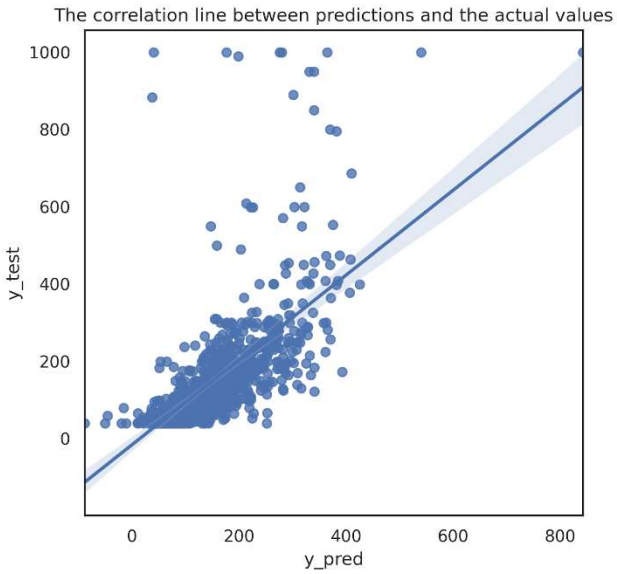



Fig 21: Ridge – Correlation plot between Predicted and Actual Values

**XGB**

```
MAE:  46.18653262011003
RMSE:  96.20211093219369
```

Fig 22: XGB – MAE and RMSE

- XGB is the best performing model with RMSE as 96.2021 (Fig 22)
- The most important features are Entire Home/Apt availability, No of bathrooms, number of people it can accommodate, gym and parking space.

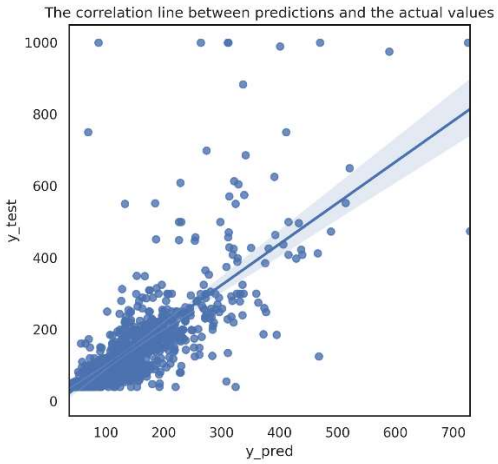



Fig 23: XGB – Correlation Plot

## Project Results:

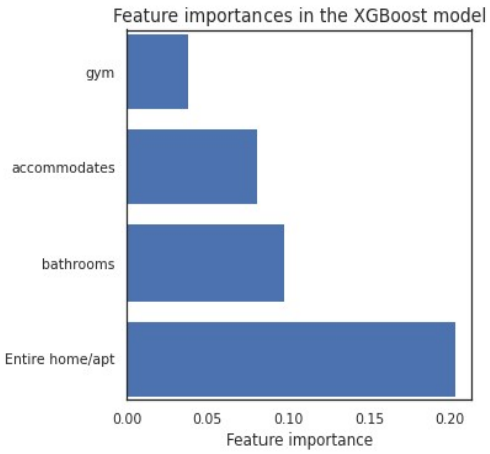



Fig 24: Important Features

- The most important features for the price of Airbnb accommodation in Boston are whether the Entire Apt/Home is available or not, Number of bathrooms, Number of people it can accommodates with feature weight as 0.20,0.09,0.08 and 0.03 respectively.
- Using XGB model, we are able to predict price with RMSE of 96.
- Few other important features are parking space availability, number of nights the Airbnb is available and host response for predicting the price.

- From project, it can also be concluded that most of the listing of the Boston city are around central Boston.

## Impact of the Project Outcomes:

Looking at the results of the different questions that we wanted to ask, we found that the In Boston mentioning features or amenities like Availability of entire apartment, Number of bathrooms, Number of apartments can accommodate and Exercise Facilities would certainly influence the price for the property. The parking space, ratings and amenities would certainly have effect as well. So, Using the model and rules of association for features would make a listing stand out from the crowd and would help hosts as well as the Airbnb to predict the best price range for any property in Boston.